

[illegible]

-3-

Synt

NTS

NTS  
NTS

NTS

NTS  
NTS

N13

NTS

NTS  
NTS

NTS

NTS  
NTS

NTS

NTS  
NTSNTS  
NTS

NTS

NTS  
NTS

NTS

NTS  
NTSNTS  
NTSNTS  
NTSNTS  
NTS

NTS

NT9

**NIH**

NTS  
NTS

NTS

NTS

NTS  
NTS

NTS

1

1

NTI

NT  
NTS

NTS

NT  
NT

NT

PI

1

1

1

1

1

2

1

RM  
VO4

.....

RRRRRRRR	MM	MM	333333	KK	KK	EEEEEEEEEE	YY	YY	DDDDDDDD	SSSSSSSS	CCCCCCCC	
RRRRRRRR	MM	MM	333333	KK	KK	EEEEEEEEEE	YY	YY	DDDDDDDD	SSSSSSSS	CCCCCCCC	
RR	RR	MMMM	33	33	KK	KK	YY	YY	DD	DD	SS	CC
RR	RR	MMMM	33	33	KK	KK	YY	YY	DD	DD	SS	CC
RR	RR	MM	33	33	KK	KK	YY	YY	DD	DD	SS	CC
RR	RR	MM	33	33	KK	KK	YY	YY	DD	DD	SS	CC
RRRRRRRR	MM	MM	33	33	KKKKKK	EEEEEEEE	YY	YY	DD	DD	SSSSSS	CC
RRRRRRRR	MM	MM	33	33	KKKKKK	EEEEEEEE	YY	YY	DD	DD	SSSSSS	CC
RR	RR	MM	33	33	KK	KK	YY	YY	DD	DD	SS	CC
RR	RR	MM	33	33	KK	KK	YY	YY	DD	DD	SS	CC
RR	RR	MM	33	33	KK	KK	YY	YY	DD	DD	SS	CC
RR	RR	MM	33	33	KK	KK	YY	YY	DD	DD	SS	CC
RR	RR	MM	33	33	KK	KK	YY	YY	DD	DD	SS	CC
RR	RR	MM	333333	KK	KK	EEEEEEEEEE	YY	YY	DDDDDDDD	SSSSSSSS	CCCCCCCC	
RR	RR	MM	333333	KK	KK	EEEEEEEEEE	YY	YY	DDDDDDDD	SSSSSSSS	CCCCCCCC	

```

LL          IIIII
LL          IIIII
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LLLLLLLLLLL IIIII
LLLLLLLLLLL IIIII
SSSSSSSS
SSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSS
SSSSSSSS

```

```
1 0001 0 MODULE RM3KEYDSC (LANGUAGE (BLISS32) ,
2 0002 0 IDENT = 'V04-000'
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1 ++
30 0030 1
31 0031 1 FACILITY: RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1 This module contains routines to allocate the key descriptors
35 0035 1
36 0036 1
37 0037 1 ENVIRONMENT:
38 0038 1
39 0039 1 VAX/VMS OPERATING SYSTEM
40 0040 1
41 0041 1 --
42 0042 1
43 0043 1
44 0044 1 AUTHOR: Wendy Koenig CREATION DATE: 27-MAR-78 9:28
45 0045 1
46 0046 1 MODIFIED BY:
47 0047 1
48 0048 1 V03-006 RAS0284 Ron Schaefer 30-Mar-1984
49 0049 1 Fix STV value on error paths for RMS$_RPL and RMS$_WPL errors.
50 0050 1
51 0051 1 V03-005 MCN0002 Maria del C. Nasr 15-Mar-1983
52 0052 1 More linkages reorganization
53 0053 1
54 0054 1 V03-004 MCN0001 Maria del C. Nasr 28-Feb-1983
55 0055 1 Reorganize linkages
56 0056 1
57 0057 1 V03-003 TMK0001 Todd M. Katz 08-Sep-1982
```



```
58 0058 1 Add support for prologue 3 SIDs. This involves correctly
59 0059 1 setting the bucket type field within each alternate key of
60 0060 1 reference index descriptor according to whether SIDR key
61 0061 1 compression is or isn't enabled.
62 0062 1
63 0063 1 V03-002 KBT0168 Keith B. Thompson 23-Aug-1982
64 0064 1 Reorganize psects
65 0065 1
66 0066 1 V03-001 KBT0057 Keith B. Thompson 9-Jun-1982
67 0067 1 Add routine rm$get_next_key and change the way key descriptors
68 0068 1 are handled
69 0069 1
70 0070 1 V02-011 PSK0003 Paulina S. Knibbe 17-Apr-1981
71 0071 1 Change the logic to initialize the bktyp fields
72 0072 1 because we are keeping track of compression status
73 0073 1 in the prologue
74 0074 1
75 0075 1 V02-010 PSK0002 Paulina S. Knibbe 10-Apr-1981
76 0076 1 Fill in the bktyp fields in the index descriptor when
77 0077 1 it is allocated and initialized
78 0078 1
79 0079 1 V02-009 PSK0001 Paulina S. Knibbe 12-Mar-1981
80 0080 1 Add datatype information to each segment in the
81 0081 1 IDX structure
82 0082 1
83 0083 1 V02-008 KPL0001 Peter Lieberwirth 12-Mar-1981
84 0084 1 Rename PSECT so branches to KEY_DESC won't break.
85 0085 1
86 0086 1 V02-007 REFORMAT Paulina S. Knibbe 23-Jul-1980
87 0087 1
88 0088 1 V0006 RAS0013 R. A. Schaefer 22-Jan-1980 14:05
89 0089 1 Change NID error to DME.
90 0090 1
91 0091 1
92 0092 1 REVISION HISTORY:
93 0093 1
94 0094 1 D. H. Gillespie, 2-AUG-78 14:31
95 0095 1 X0002 - add one long word to in core key descriptor containing area numbers
96 0096 1
97 0097 1 Wendy Koenig, 3-AUG-78 12:38
98 0098 1 X0003 - ACCESS KEY DESCRIPTORS DIRECTLY, RATHER THAN THRU VBN 1 LINKS
99 0099 1
100 0100 1 Wendy Koenig, 24-OCT-78 14:02
101 0101 1 X0004 - MAKE CHANGES CAUSED BY SHARING CONVENTIONS
102 0102 1
103 0103 1 Wendy Koenig, 6-NOV-78 13:04
104 0104 1 X0005 - MAKE PLG ERRORS INTO RPL
105 0105 1
106 0106 1 *****
107 0107 1
108 0108 1 LIBRARY 'RMSLIB:RMS';
109 0109 1
110 0110 1 REQUIRE 'RMSSRC:RMSIDXDEF';
111 0175 1
112 0176 1 Define default psects for code
113 0177 1
114 0178 1 PSECT
```

```
115 0179 1 CODE = RMSRMS3(PSECT_ATTR),
116 0180 1 PLIT = RMSRMS3(PSECT_ATTR);
117 0181 1
118 0182 1 ! Define a local linkage
119 0183 1 !
120 0184 1 MACRO
121 M 0185 1 L_FILL_IN =
122 M 0186 1 RL$FILL_IN = JSB(REGISTER=6) : GLOBAL(R_IFAB,R_IDX_DFN) NOTUSED(8,9)
123 M 0187 1 NOPRESERVE(2,3,6)
124 0188 1 %;
125 0189 1
126 0190 1 ! Define linkage
127 0191 1 !
128 0192 1 LINKAGE
129 0193 1 L_CACHE,
130 0194 1 L_CHKSUM,
131 0195 1 L_FILL_IN,
132 0196 1 L_GETSPC,
133 0197 1 L_LINK_7_10_11,
134 0198 1 L_RABREG_7,
135 0199 1 L_RELEASE;
136 0200 1
137 0201 1 ! External routines
138 0202 1 !
139 0203 1 EXTERNAL ROUTINE
140 0204 1 RMS$CACHE : RL$CACHE,
141 0205 1 RMS$CHKSUM : RL$CHKSUM,
142 0206 1 RMS$GETBLK : RL$GETSPC,
143 0207 1 RMS$RELEASE : RL$RELEASE;
144 0208 1
145 0209 1 ! Forward routines
146 0210 1 !
147 0211 1 FORWARD ROUTINE
148 0212 1 FILL_IN : RL$FILL_IN,
149 0213 1 RMS$AC_KEY_DESC : RL$LINK_7_10_11,
150 0214 1 RMS$KEY_DESC : RL$RABREG_7,
151 0215 1 RMS$GET_NEXT_KEY : RL$LINK_7_10_11;
152 0216 1
153 0217 1 ! Define some local macros
154 0218 1 !
155 0219 1 MACRO
156 M 0220 1 KEYOFFSET (SYMBOL, OFF) =
157 M 0221 1 $BYTEOFFSET(SYMBOL)+(OFF*($BYTESIZE(SYMBOL))),
158 M 0222 1 $BITPOSITION(SYMBOL),
159 M 0223 1 $FIELDWIDTH(SYMBOL),
160 M 0224 1 $EXTENSION(SYMBOL)
161 0225 1 %,
162 M 0226 1 POSITIONMAC (OFF) =
163 M 0227 1 OFF,$BITPOSITION(IDX$W_POSITION),
164 M 0228 1 $FIELDWIDTH(IDX$W_POSITION),$EXTENSION(IDX$W_POSITION)
165 0229 1 %,
166 M 0230 1 SIZEMAC (OFF) =
167 M 0231 1 OFF+2,$BITPOSITION(IDX$B_SIZE),
168 M 0232 1 $FIELDWIDTH(IDX$B_SIZE),$EXTENSION(IDX$B_SIZE)
169 0233 1 %,
170 M 0234 1 TYPEMAC (OFF) =
171 M 0235 1 OFF+3,$BITPOSITION(IDX$B_TYPE),
```

RM3KEYDSC  
V04-000

L 11  
16-Sep-1984 01:49:04  
14-Sep-1984 13:01:27

VAX-11 Bliss-32 V4.0-742  
[RMS.SRC]RM3KEYDSC.B32;1

Page 4  
(1)

: 172  
: 173  
: 174

M 0236 1  
0237 1 x:  
0238 1

\$FIELDWIDTH(IDX\$B\_TYPE), \$EXTENSION(IDX\$B\_TYPE)

RM  
VO



```
176 0239 1 %SBTTL 'FILL_IN'
177 0240 1 ROUTINE FILL_IN ( DESC ) : RLS$FILL_IN =
178 0241 1
179 0242 1 ++
180 0243 1
181 0244 1 FILL_IN
182 0245 1
183 0246 1
184 0247 1 CALLING SEQUENCE:
185 0248 1
186 0249 1 FILL_IN (DESC)
187 0250 1
188 0251 1 INPUT PARAMETERS:
189 0252 1
190 0253 1 DESC is the address of the key descriptor in prologue
191 0254 1
192 0255 1 IMPLICIT INPUTS:
193 0256 1
194 0257 1 R7 -- INDEX DESCRIPTOR address
195 0258 1
196 0259 1 OUTPUT PARAMETERS:
197 0260 1 none
198 0261 1
199 0262 1 IMPLICIT OUTPUTS:
200 0263 1 none
201 0264 1
202 0265 1 ROUTINE VALUE:
203 0266 1
204 0267 1 always RMSSUC
205 0268 1
206 0269 1 SIDE EFFECTS:
207 0270 1
208 0271 1 fills in the index descriptor
209 0272 1
210 0273 1 --
211 0274 1
212 0275 2 BEGIN
213 0276 2
214 0277 2 EXTERNAL REGISTER
215 0278 2 R_IFAB_STR,
216 0279 2 R_IDX_DFN_STR;
217 0280 2
218 0281 2 MAP
219 0282 2 DESC : REF BBLOCK;
220 0283 2
221 0284 2 LITERAL
222 0285 2 BEGINNING_IDX = $BYTEOFFSET(Idx$B_IANUM),
223 0286 2 BEGINNING_KEY = $BYTEOFFSET(KEY$B_IANUM),
224 0287 2 POSOFFSET = $BYTEOFFSET(Idx$W_POSITION);
225 0288 2
226 0289 2 DECR I FROM .DESC [ KEY$B_SEGMENTS ] - 1 TO 0 DO
227 0290 2 BEGIN
228 0291 2
229 0292 2
230 0293 2
231 0294 2 IDX_DFN [ POSITIONMAC( POSOFFSET + ( 4 * .I ) ) ] =
232 0295 2 .DESC [ KEYOFFSET( %QUOTE KEY$W_POSITION,.I ) ];
```

```
233      0296      3      IDX_DFN [ SIZEMAC( POSOFFSET + ( 4 * .1 ) ) ] =
234      0297      3      .DESC [ KEYOFFSET( %QUOTE KEY$B_SIZE, .1 ) ];
235      0298      3      IDX_DFN [ TYPEMAC( POSOFFSET + ( 4 * .1 ) ) ] =
236      0299      3      KEY$C_STRING;
237      0300      3      END;
238      0301      3
239      0302      3      IF .IFAB [ IFB$B_PLG_VER ] GTR PLG$C_VER_IDX
240      0303      3      THEN
241      0304      3
242      0305      3          ! Load up the datatype fields for each segment (potentially
243      0306      3          ! different)
244      0307      3
245      0308      3      BEGIN
246      0309      3      DECR I FROM .DESC [ KEY$B_SEGMENTS ] - 1 TO 0 DO
247      0310      3
248      0311      3          IDX_DFN [ TYPEMAC( POSOFFSET + ( 4 * .1 ) ) ] =
249      0312      3          .DESC [ KEYOFFSET( %QUOTE KEY$B_TYPE, .1 ) ];
250      0313      3      END
251      0314      3      ELSE
252      0315      3
253      0316      3          ! Fix up the first datatype to be correct (in case this
254      0317      3          ! wasn't a segmented key.
255      0318      3
256      0319      3          IDX_DFN [ TYPEMAC( POSOFFSET ) ] = .DESC [ KEY$B_DATATYPE ];
257      0320      3
258      0321      3      CH$MOVE( IDX$C_FIXED_BLN - BEGINNING_IDX,
259      0322      3          .DESC + BEGINNING_KEY,
260      0323      3          .IDX_DFN + BEGINNING_IDX );
261      0324      3
262      0325      3      ! Fill in the bucket types for this index
263      0326      3
264      0327      3      IF .IFAB [ IFB$B_PLG_VER ] LSSU PLG$C_VER_3
265      0328      3      THEN
266      0329      3      BEGIN
267      0330      3          IDX_DFN [ IDX$B_DATBKTYT ] = IDX$C_V2_BKT;
268      0331      3          IDX_DFN [ IDX$B_IDXBKTYT ] = IDX$C_V2_BKT;
269      0332      3      END
270      0333      3      ELSE
271      0334      3
272      0335      3          ! First initialize the data bucket type (if
273      0336      3          ! this is the primary key index)
274      0337      3
275      0338      3      BEGIN
276      0339      3
277      0340      3          IF .IDX_DFN [ IDX$B_KEYREF ] EQL 0
278      0341      3          THEN
279      0342      3
280      0343      3              IF .IDX_DFN [ IDX$V_REC_COMPR ]
281      0344      3              THEN
282      0345      3
283      0346      3                  IF .IDX_DFN [ IDX$V_KEY_COMPR ]
284      0347      3                  THEN
285      0348      3
286      0349      3                      ! Primary key is compressed, data is compressed
287      0350      3
288      0351      3                      IDX_DFN [ IDX$B_DATBKTYT ] = IDX$C_CMPCMP
289      0352      3
```



```

: 290      0353      3      ELSE
: 291      0354      3
: 292      0355      3      ! Primary key is not compressed, data is compressed
: 293      0356      3
: 294      0357      3      IDX_DFN [ IDX$B_DATBKTY ] = IDX$C_NCMPCMP
: 295      0358      3
: 296      0359      3      ELSE
: 297      0360      3
: 298      0361      3      IF .IDX_DFN [ IDX$V_KEY_COMPR ]
: 299      0362      3      THEN
: 300      0363      3
: 301      0364      3      ! Primary key is compressed, data is not compressed
: 302      0365      3
: 303      0366      3      IDX_DFN [ IDX$B_DATBKTY ] = IDX$C_CMPNCMP
: 304      0367      3
: 305      0368      3      ELSE
: 306      0369      3
: 307      0370      3      ! Primary key is not compressed, data is not compressed
: 308      0371      3
: 309      0372      3      IDX_DFN [ IDX$B_DATBKTY ] = IDX$C_NCMPNCMP
: 310      0373      3
: 311      0374      3      ! Otherwise, this must be an alternate key of reference. Initialize
: 312      0375      3      the SDR bucket type.
: 313      0376      3
: 314      0377      3      ELSE
: 315      0378      3      IF .IDX_DFN[IDX$V_KEY_COMPR]
: 316      0379      3      THEN
: 317      0380      3      IDX_DFN[IDX$B_DATBKTY] = IDX$C_CMPCMP
: 318      0381      3      ELSE
: 319      0382      3      IDX_DFN[IDX$B_DATBKTY] = IDX$C_NCMPNCMP;
: 320      0383      3
: 321      0384      3      ! Now initialize the index bucket type
: 322      0385      3
: 323      0386      3      IF .IDX_DFN [ IDX$V_IDX_COMPR ]
: 324      0387      3      THEN
: 325      0388      3      IDX_DFN [ IDX$B_IDXBKTY ] = IDX$C_CMPIDX
: 326      0389      3
: 327      0390      3      ELSE
: 328      0391      3      IDX_DFN [ IDX$B_IDXBKTY ] = IDX$C_NCMPIDX;
: 329      0392      3
: 330      0393      3      END;
: 331      0394      3      RETURN RMSSUC()
: 332      0395      3
: 333      0396      3      END;
```

```

.TITLE RM3KEYDSC
.IDENT \V04-000\

.EXTRN RM$CACHE, RM$CHKSUM
.EXTRN RM$GETBLK, RM$RELEASE

.PSECT RM$RMS3,NOWRT, GBL, PIC,2
```

```

50      12      30      BB 00000 FILL_IN: PUSHR #^M<R4,R5>
          A6      9A 00002      MOVZBL 18(DESC), 1
          18      11 00006      BRB 2$
```

```

: 0240
: 0294
:
```

			2C	A740	DF	00008	1\$:	PUSHAL	44(IDX_DFN)[1]		0295
9E			1C	A640	B0	0000C		MOVW	28(DESC)[1], @ (SP)+		
			2E	A740	DF	00011		PUSHAL	46(IDX_DFN)[1]		0297
9E			2C	A046	90	00015		MOVB	44(I)[DESC], @ (SP)+		
			2F	A740	DF	0001A		PUSHAL	47(IDX_DFN)[1]		0298
				9E	94	0001E		CLRB	@ (SP)+		
E5				50	F4	00020	2\$:	SOBGEQ	I, 1\$		0289
02			00B7	CA	91	00023		CMPB	183(IFAB), #2		0302
				14	1B	00028		BLEQU	5\$		
50			12	A6	9A	0002A		MOVZBL	18(DESC), I		0309
				09	11	0002E		BRB	4\$		
			2F	A740	DF	00030	3\$:	PUSHAL	47(IDX_DFN)[1]		0312
9E			58	A046	90	00034		MOVB	88(I)[DESC], @ (SP)+		
F4				50	F4	00039	4\$:	SOBGEQ	I, 3\$		0311
				05	11	0003C		BRB	6\$		0302
		2F	A7	11	A6	90	0003E	5\$:	MOVB	17(DESC), 47(IDX_DFN)	0319
12	A7	06	A6		1A	28	00043	6\$:	MOV C3	#26, 6(DESC), 18(IDX_DFN)	0323
			50		29	A7	9E	00049	MOVAB	41(IDX_DFN), R0	0330
			03		00B7	CA	91	0004D	CMPB	183(IFAB), #3	0327
						07	1E	00052	BGEQU	7\$	
						60	94	00054	CLRB	(R0)	0330
			28	A7	94	00056		CLRB	40(IDX_DFN)		0331
				39	11	00059		BRB	14\$		0327
		51	1C	A7	9E	0005B	7\$:	MOVAB	28(IDX_DFN), R1		0343
			21	A7	95	0005F		TSTB	33(IDX_DFN)		0340
				16	12	00062		BNEQ	9\$		
				61	95	00064		TSTB	(R1)		0343
				09	18	00066		BGEQ	8\$		
	12		61		06	E0	00068	BBS	#6, (R1), 10\$		0346
			60		05	90	0006C	MOVB	#5, (R0)		0357
					15	11	0006F	BRB	12\$		0346
	0E		61		06	E1	00071	8\$:	BBC	#6, (R1), 11\$	0361
			60		04	90	00075		MOVB	#4, (R0)	0366
					0C	11	00078	BRB	12\$		
	05		61		06	E1	0007A	9\$:	BBC	#6, (R1), 11\$	0378
			60		03	90	0007E	10\$:	MOVB	#3, (R0)	0380
					03	11	00081	BRB	12\$		
			60		06	90	00083	11\$:	MOVB	#6, (R0)	0382
	06		61		03	E1	00086	12\$:	BBC	#3, (R1), 13\$	0386
		28	A7		01	90	0008A		MOVB	#1, 40(IDX_DFN)	0388
					04	11	0008E	BRB	14\$		
		28	A7		02	90	00090	13\$:	MOVB	#2, 40(IDX_DFN)	0390
			50		01	D0	00094	14\$:	MOVL	#1, R0	0394
					30	BA	00097		POPR	#M<R4,R5>	0396
					05	00099		RSB			

; Routine Size: 154 bytes, Routine Base: RMSRMS3 + 0000

; 334 0397 1

## RMSAL\_KEY\_DESC

```
0398 1 XSBTTL 'RMSAL_KEY_DESC'
0399 1 GLOBAL ROUTINE RMSAL_KEY_DESC ( DESC,VBN,OFFSET ) : RL$LINK_7_10_11 =
0400 1
0401 1 ++
0402 1
0403 1 RMSAL_KEY_DESC
0404 1
0405 1 This routine allocates the in-memory index descriptor,
0406 1 links it in, and fills it in.
0407 1
0408 1 The format of the index descriptor list must be as follows:
0409 1
0410 1 o The first index descriptor is pointer to by ifb$I_idx_ptr
0411 1
0412 1 o The index descriptors are linked through idx$I_idxfl
0413 1
0414 1 o The last index descriptor has idx$I_idxfl set to zero
0415 1
0416 1 o The primary key descriptor (key_ref = 0) must be the
0417 1 first descriptor in the list and have a descriptor
0418 1 number of zero (i.e. idx$b_desc_no = 0).
0419 1
0420 1 o Non primary key descriptors may appear in any order
0421 1 by key of reference.
0422 1
0423 1 CALLING SEQUENCE:
0424 1
0425 1 RMSAL_KEY_DESC ( desc,vbn,offset )
0426 1
0427 1 INPUT PARAMETERS:
0428 1
0429 1 desc - pointer to descriptor in prologue
0430 1 vbn - VBN where the descriptor is in
0431 1 offset - byte offset in the VBN where the descriptor is
0432 1
0433 1 IMPLICIT INPUTS:
0434 1
0435 1 R10 IFAB address
0436 1 R11 IMPURE AREA address
0437 1
0438 1 OUTPUT PARAMETERS:
0439 1 none
0440 1
0441 1 IMPLICIT OUTPUTS:
0442 1
0443 1 The address of the index descriptor is returned in IDX_DFN
0444 1
0445 1 ROUTINE VALUE:
0446 1
0447 1 Standard rms, in particular SUC, DME
0448 1
0449 1 SIDE EFFECTS:
0450 1
0451 1 The index descriptor is allocated, linked in, filled in
0452 1
0453 1 --
0454 1
```



```
393 0455 2 BEGIN
394 0456
395 0457 ! Define external registers
396 0458
397 0459 EXTERNAL REGISTER
398 0460 R_IDX_DFN_STR,
399 0461 R_IMPURE_STR,
400 0462 R_IFAB_STR;
401 0463
402 0464 MAP
403 0465 DESC : REF BBLOCK; ! desc points to a key descriptor
404 0466
405 0467 ! if the key's datatype is illegal return an error
406 0468
407 0469 IF .DESC [ KEYSB_DATATYPE ] GTRU KEYSB_MAX_DATA
408 0470 THEN
409 0471 RETURN RMSERR( DTP );
410 0472
411 0473 BEGIN
412 0474
413 0475 LOCAL
414 0476 BLK : REF BBLOCK;
415 0477
416 0478 ! Size of index descriptor = fixed portion in longwords + variable portion
417 0479 ! in longwords which is 1 longword per segment (i.e. 1 word per size and 1
418 0480 ! word per position)
419 0481
420 0482 IF NOT RMSGETBLK( .IFAB,
421 0483 IDXSC_FIXED_BLN / 4 + .DESC [ KEYSB_SEGMENTS ];
422 0484 BLK )
423 0485 THEN
424 0486 RETURN RMSERR( DME );
425 0487
426 0488 IDX_DFN = .BLK
427 0489
428 0490 END;
429 0491
430 0492 BEGIN
431 0493
432 0494 LOCAL
433 0495 NUMBER : INITIAL(0);
434 0496
435 0497 ! We now have an index descriptor in IDX_DFN, size has been filled in
436 0498 ! link it in. NOTE: Key descriptor 0 must be at the head of the list
437 0499
438 0500 IDX_DFN [ IDXSL_IDXFL ] = 0;
439 0501
440 0502 ! If there are none then link it in the front
441 0503
442 0504 IF .IFAB [ IFBSL_IDX_PTR ] EQL 0
443 0505 THEN
444 0506 BEGIN
445 0507
446 0508 IFAB [ IFBSL_IDX_PTR ] = .IDX_DFN;
447 0509
448 0510 ! If this is not the primary key then make the descriptor non-zero
449 0511
```

```

450      0512 4      IF .DESC [ KEYSB_KEYREF ] NEQ 0
451      0513 4      THEN
452      0514 4          NUMBER = 1
453      0515 4
454      0516 4      END
455      0517 4      ELSE
456      0518 4          ! If this is key 0 it must go to the head of the list
457      0519 4          ! else put it at the end of the chain
458      0520 4
459      0521 4      IF .DESC [ KEYSB_KEYREF ] EQL 0
460      0522 4      THEN
461      0523 4          BEGIN
462      0524 4              IDX_DFN [ IDXSL_IDXFL ] = .IFAB [ IFBSL_IDX_PTR ];
463      0525 4              IFAB [ IFBSL_IDX_PTR ] = .IDX_DFN
464      0526 4          END
465      0527 4      ELSE
466      0528 4          BEGIN
467      0529 4              LOCAL
468      0530 4                  PTR      : REF BBLOCK;
469      0531 4
470      0532 4                  PTR = .IFAB [ IFBSL_IDX_PTR ];
471      0533 4
472      0534 4                  ! Find the last index descriptor
473      0535 4                  !
474      0536 4                  ! Find the last index descriptor
475      0537 4                  !
476      0538 4                  WHILE .PTR [ IDXSL_IDXFL ] NEQ 0
477      0539 4                  DO
478      0540 4                      PTR = .PTR [ IDXSL_IDXFL ];
479      0541 4
480      0542 4                  ! The number of this descriptor is one higher then the last one in
481      0543 4                  ! the chain
482      0544 4                  !
483      0545 4                  NUMBER = .PTR [ IDXSB_DESC_NO ] + 1;
484      0546 4
485      0547 4                  ! Place the new descriptor at the end of the chain
486      0548 4                  !
487      0549 4                  PTR [ IDXSL_IDXFL ] = .IDX_DFN
488      0550 4
489      0551 4                  END;
490      0552 4
491      0553 4          ! Now fill it in
492      0554 4          !
493      0555 4          !
494      0556 4          !
495      0557 4          !
496      0558 4          !
497      0559 4          !
498      0560 4          !
499      0561 4          !
500      0562 4          !
501      0563 4          !
502      0564 4          !

```

		005C	BF	BB	00000	RMSAL_KEY_DESC::	
56	14	AE	D0	00004		PUSHR #M<R2,R3,R4,R6>	0399
07	11	A6	91	00008		MOVL DESC, R6	0469
		07	1B	0000C		CMPB 17(R6), #7	
50	84E4	BF	3C	0000E		BLEQU 1\$	0471
		63	11	00013		MOVZWL #34020, R0	
52	12	A6	9A	00015	1\$:	BRB 7\$	0483
52		0B	C0	00019		MOVZBL 18(R6), R2	
51		5A	D0	0001C		ADDL2 #11, R2	0482
		0000G	30	0001F		MOVL IFAB, R1	
07		50	E8	00022		BSBW RMSGETBLK	
50	84D4	BF	3C	00025		BLBS R0, 2\$	0486
		4C	11	0002A		MOVZWL #34004, R0	
57		51	D0	0002C	2\$:	BRB 7\$	0488
		51	D4	0002F		MOVL BLK, IDX_DFN	0492
		67	D4	00031		CLRL NUMBER	0500
50	00AC	CA	9E	00033		CLRL (IDX_DFN)	0504
		60	D5	00038		MOVAB 172(IFAB), R0	
		0D	12	0003A		TSTL (R0)	
60	15	57	D0	0003C		BNEQ 3\$	0508
		1F	13	00042		MOVL IDX_DFN, (R0)	0512
51		01	D0	00044		TSTB 21(R6)	
		1A	11	00047		BEQL 6\$	0514
	15	A6	95	00049	3\$:	MOVL #1, NUMBER	0512
		05	12	0004C		BRB 6\$	0522
67		60	D0	0004E		TSTB 21(R6)	
		0D	11	00051		BNEQ 4\$	
50		60	D0	00053	4\$:	MOVL (R0), (IDX_DFN)	0525
		60	D5	00056		BRB 5\$	0526
		F9	12	00058		MOVL (R0), PTR	0534
51	10	A0	9A	0005A		TSTL (PTR)	0538
		51	D6	0005E		BNEQ 4\$	
60		57	D0	00060	5\$:	MOVZBL 16(PTR), NUMBER	0545
0A	A7	18	AE	00063	6\$:	INCL NUMBER	
0E	A7	1C	AE	00068		MOVL IDX_DFN, (PTR)	0549
10	A7		51	0006D		MOVL VBN, 10(IDX_DFN)	0555
08	A7		0F	00071		MOVW OFFSET, 14(IDX_DFN)	0556
		FEEE	30	00075		MOVB NUMBER, 16(IDX_DFN)	0557
		8F	BA	00078	7\$:	MOVB #15, 8(IDX_DFN)	0558
		05	0007C			BSBW FILL IN	0562
						POPR #M<R2,R3,R4,R6>	0564
						RSB	

; Routine Size: 125 bytes, Routine Base: RMSRMS3 + 009A

; 503 0565 1



## RMSKEY\_DESC

```
505 0566 1 %SBTTL 'RMSKEY_DESC'
506 0567 1 GLOBAL ROUTINE RMSKEY_DESC (KEYREF) : RLSRABREG_7 =
507 0568 1
508 0569 1 ++
509 0570 1
510 0571 1 RMSKEY_DESC
511 0572 1
512 0573 1 Given the key of reference, this routine sets idx_dfn to the correct
513 0574 1 index descriptor address. It searches the existing index descriptors
514 0575 1 for a match and, if it does not find it, allocates it
515 0576 1
516 0577 1 EXCEPTION: if NEW_IDX is set in the irab, and if the index descriptor
517 0578 1 already exists, fill it in again ( but don't re-allocate it)
518 0579 1
519 0580 1 CALLING SEQUENCE:
520 0581 1 RMSKEY_DESC (KEYREF)
521 0582 1
522 0583 1 INPUT PARAMETERS:
523 0584 1
524 0585 1 keyref = key of reference
525 0586 1
526 0587 1 IMPLICIT INPUTS:
527 0588 1
528 0589 1 R8 -- RAB address
529 0590 1 R9 -- IRAB address
530 0591 1 R10 -- IFAB address
531 0592 1 R11 -- IMPURE AREA address
532 0593 1 NEW_IDX, CACHEFLGS in IRAB
533 0594 1
534 0595 1 OUTPUT PARAMETERS:
535 0596 1 none
536 0597 1
537 0598 1 IMPLICIT OUTPUTS:
538 0599 1
539 0600 1
540 0601 1 IDX_DFN will contain the address of the index descriptor
541 0602 1 NEW_IDX is cleared
542 0603 1 CACHEFLGS is cleared
543 0604 1 If CACHEFLGS was non-zero ( i.e. the block was locked),
544 0605 1 lock_bdb contains the bdb associated w/ the block
545 0606 1
546 0607 1 ROUTINE VALUE:
547 0608 1
548 0609 1 usual rms status codes, particularly SUC,KRF,RPL
549 0610 1 and those returned by RMSAL_KEY_DESC
550 0611 1
551 0612 1 SIDE EFFECTS:
552 0613 1
553 0614 1 Allocates idx descriptor if it doesn't exist, fills it in & links it in
554 0615 1 may lock up the block containing the key descriptor
555 0616 1
556 0617 1 --
557 0618 1
558 0619 2 BEGIN
559 0620 2
560 0621 2 EXTERNAL REGISTER
561 0622 2 R_IDX_DFN_STR,
```

```
562      COMMON_RAB_STR;  
563  
564      GLOBAL REGISTER  
565      COMMON_IO_STR;  
566  
567      MAP  
568      KEYREF : BYTE;  
569  
570      LOCAL  
571      STATUS;  
572  
573      ! Find the index descriptor and return its address in IDX_DFN  
574      !  
575      IDX_DFN = .IFAB [ IFBSL_IDX_PTR ];  
576  
577      WHILE .IDX_DFN [ IDX$B_KEYREF ] NEQ .KEYREF  
578      DO  
579  
580          ! If this is the last key then the key does not exist  
581          !  
582          IF ( IDX_DFN = .IDX_DFN [ IDX$B_IDXFL ] ) EQL 0  
583          THEN  
584              BEGIN  
585                  IRAB [ IRB$V_NEW_IDX ] = 0;  
586                  IRAB [ IRB$B_CACHEFLGS ] = 0;  
587                  RETURN RMSERR( KRF )  
588              END;  
589  
590          ! If we don't have to restuff the descriptor simply return  
591          !  
592          IF NOT .IRAB [ IRB$V_NEW_IDX ]  
593          THEN  
594              BEGIN  
595                  IRAB [ IRB$B_CACHEFLGS ] = 0;  
596                  RETURN RMSSUC()  
597              END;  
598  
599          ! We clear NEW_IDX  
600          !  
601          IRAB [ IRB$V_NEW_IDX ] = 0;  
602  
603          ! Go get the block  
604          !  
605          RETURN_ON_ERROR( RMSCACHE( .IDX_DFN [ IDX$B_VBN ], 512, .IRAB [ IRB$B_CACHEFLGS ] ),  
606          BEGIN  
607              IRAB [ IRB$B_CACHEFLGS ] = 0;  
608              IF .RAB [ RAB$B_STV ] EQL 0  
609              THEN  
610                  RAB [ RAB$B_STV ] = .STATUS OR 1*16;  
611                  STATUS = RMSERR( RPL )  
612              END );  
613  
614          ! If the chksum is bad, release the block and return  
615          !  
616          RETURN_ON_ERROR( RM$CHKSUM(),  
617          BEGIN  
618              IRAB [ IRB$B_CACHEFLGS ] = 0;
```

```

P 0680 2
0681
0682
0683
0684
0685
0686
0687
0688
0689
0690
0691
0692
0693
0694
0695
0696
0697
0698
0699
0700 1

RMSRELEASE(0)
END );

! Fill in the descriptor with the fresh copy
STATUS = FILL_IN( .BKT_ADDR + .IDX_DFN [ IDX$W_OFFSET ] );

! If the block wasn't locked, release it otherwise, it is up to the caller
! to release it if this is the case, set up lock_bdb to point to it
IF .IRAB [ IRB$B_CACHEFLGS ] EQL 0
THEN
  RMSRELEASE(0)
ELSE
  IRAB [ IRB$L_LOCK_BDB ] = .BDB;

IRAB [ IRB$B_CACHEFLGS ] = 0;

RETURN .STATUS

END;
```

		007C	8F	BB	00000	RMSKEY_DESC::		
						PUSHR	#^M<R2,R3,R4,R5,R6>	0567
						MOVL	172(IFAB), IDX_DFN	0636
18	AE	00AC	CA	D0	00004	CMPB	33(IDX_DFN), KEYREF	0638
		21	A7	91	00009	BEQL	2\$	
	57		67	D0	00010	MOVL	(IDX_DFN), IDX_DFN	0643
			F4	12	00013	BNEQ	1\$	
40	A9	000800FF	8F	CA	00015	BICL2	#524543, 64(IRAB)	0646
	50	859C	8F	3C	0001D	MOVZWL	#34204, R0	0648
			77	11	00022	BRB	9\$	
08	42	A9	03	E0	00024	BBS	#3, 66(IRAB), 3\$	0653
		40	A9	94	00029	CLRB	64(IRAB)	0656
	50		01	D0	0002C	MOVL	#1, R0	0657
			6A	11	0002F	BRB	9\$	
42	A9		08	8A	00031	BICB2	#8, 66(IRAB)	0662
	53	40	A9	9A	00035	MOVZBL	64(IRAB), R3	0673
	52	0200	8F	3C	00039	MOVZWL	#512, R2	
	51	0A	A7	D0	0003E	MOVL	10(IDX_DFN), R1	
			0000G	30	00042	BSBW	RMS\$CACHE	
	18		50	E8	00045	BLBS	STATUS, 5\$	
		40	A9	94	00048	CLRB	64(IRAB)	
		0C	A8	D5	0004B	TSTL	12(RAB)	
			09	12	0004E	BNEQ	4\$	
0C	A8	50	00010000	8F	C9	BISL3	#65536, STATUS, 12(RAB)	
		50	C104	8F	3C	MOVZWL	#49412, STATUS	
			3B	11	0005E	BRB	9\$	
			0000G	30	00060	BSBW	RMS\$CHKSUM	0681
	56		50	D0	00063	MOVL	R0, STATUS	
	0D		56	E8	00066	BLBS	STATUS, 6\$	
		40	A9	94	00069	CLRB	64(IRAB)	
			53	D4	0006C	CLRL	R3	



RM3KEYDSC  
V04-000

RMSKEY\_DESC

K 12  
16-Sep-1984 01:49:04  
14-Sep-1984 13:01:07

VAX-11 Bliss-32 V4.0-742  
[RMS.SRC]RM3KEYDSC.B32;1

Page 16  
(4)

56

50	0000G	30	0006E	BSBW	RMSRELEASE
	56	D0	00071	MOVL	STATUS, R0
	25	11	00074	BI B	9\$
50	0E	A7	3C 00076	MO'ZWL	14(IDX DFN), R0
55		50	C1 0007A	ADD.3	R0, BKT_ADDR, R6
	FE68	30	0007E	BSBW	FILL IN
55		50	D0 00081	MOVL	R0, STATUS
	40	A9	95 00084	TSTB	64(IRAB)
		07	12 00087	BNEQ	7\$
		53	D4 00089	CLRL	R3
	0000G	30	0008B	BSBW	RMSRELEASE
	05	11	0008E	BRB	8\$
0084	C9	54	D0 00090	MOVL	BDB, 132(IRAB)
		40	A9 94 00095	CLRB	64(IRAB)
50		55	D0 00098	MOVL	STATUS, R0
	007C	8F	BA 0009B	POPR	#*M<R2,R3,R4,R5,R6>
		05	0009F	RSB	

0685

0690

0692

0694

0696

0698

0700

; Routine Size: 160 bytes, Routine Base: RMSRMS3 + 0117

; 640 0701 1

## RMSGET\_NEXT\_KEY

```
642 0702 1 %SBTTL 'RMSGET_NEXT_KEY'
643 0703 1 GLOBAL ROUTINE 'RMSGET_NEXT_KEY : RLSLINK_7_10_11 =
644 0704 1
645 0705 1 !++
646 0706 1
647 0707 1 RMSGET_NEXT_KEY
648 0708 1
649 0709 1     Sets idx_dfn to the address of the next key descriptor if there is one
650 0710 1     Else it leaves idx_dfn alone
651 0711 1
652 0712 1 CALLING SEQUENCE:
653 0713 1
654 0714 1     RMSGET_NEXT_KEY()
655 0715 1
656 0716 1 INPUT PARAMETERS:
657 0717 1     none
658 0718 1
659 0719 1 IMPLICIT INPUTS:
660 0720 1
661 0721 1     idx_dfn - current index descriptor
662 0722 1
663 0723 1 OUTPUT PARAMETERS:
664 0724 1     none
665 0725 1
666 0726 1 IMPLICIT OUTPUTS:
667 0727 1
668 0728 1     idx_dfn - will contain the address of the next index descriptor if
669 0729 1     there is one otherwise it is not affected
670 0730 1
671 0731 1 ROUTINE VALUE:
672 0732 1
673 0733 1     1 - there was a next index descriptor
674 0734 1     0 - there was not a next one
675 0735 1
676 0736 1 SIDE EFFECTS:
677 0737 1     none
678 0738 1
679 0739 1 --
680 0740 1
681 0741 2 BEGIN
682 0742 2
683 0743 2 EXTERNAL REGISTER
684 0744 2     R_IDX_DFN_STR;
685 0745 2
686 0746 2 ! If there isn't anymore index descriptors then exit
687 0747 2 !
688 0748 2 IF .IDX_DFN [ IDX$$_IDXFL ] EQL 0
689 0749 2 THEN
690 0750 2     RETURN 0;
691 0751 2
692 0752 2     IDX_DFN = .IDX_DFN [ IDX$$_IDXFL ];
693 0753 2
694 0754 2 RETURN 1
695 0755 2
696 0756 1 END;
```

```

67 D5 00000 RM$GET_NEXT_KEY::
                                TSTC (IDX_DFN)
07 13 00002                                BEQL 1$
67 D0 00004                                MOVL (IDX_DFN), IDX_DFN
01 D0 00007                                MOVL #1, R0
                                RSB
50 D4 0000B 1$: CLRL R0
                                RSB
05 0000D

```

```

: 0748
: 0752
: 0754
: 0756
:

```

; Routine Size: 14 bytes, Routine Base: RM\$RMS3 + 01B7

```

: 697      0757 1
: 698      0758 1 END
: 699      0759 1
: 700      0760 0 ELUDOM

```

# PSECT SUMMARY

Name	Bytes	Attributes
RM\$RMS3	453	NOVEC,NOWRT, RD , EXE,NOSHR, GBL, REL, CON, PIC,ALIGN(2)

# Library Statistics

File	Symbols		Percent	Pages Mapped	Processing Time
	Total	Loaded			
_\$255\$DUA28:[RMS.OBJ]RMS.L32;1	3109	79	2	154	00:00.4

# COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:RM3KEYDSC/OBJ=OBJ\$:RM3KEYDSC MSRC\$:RM3KEYDSC/UPDATE=(ENH\$:RM3KEYDSC)

```

; Size:      453 code + 0 data bytes
; Run Time:   00:13.1
; Elapsed Time: 00:28.0
; Lines/CPU Min: 3486
; Lexemes/CPU-Min: 21307
; Memory Used: 103 pages

```



RM3KEYDSC  
V04-000

RMSGET\_NEXT\_KEY

; Compilation Complete

N 12  
16-Sep-1984 01:49:04

VAX-11 Bliss-32 V4.0-742

Page 19

RM3  
V04



0325 AH-BT13A-SE  
VAX/VMS V4.0

**DIGITAL EQUIPMENT CORPORATION**  
**CONFIDENTIAL AND PROPRIETARY**